Parallel Transport System Design

Torsten Braun[a] and Martina Zitterbart[b]

[a] Institute of Telematics, University of Karlsruhe, P.O. Box 6980, Zirkel 2, W-7500 Karlsruhe, Germany

[b] IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, U.S.A. (currently on leave from the University of Karlsruhe, Germany)

Abstract

Requirements of emerging applications together with rapid advances in networking technology towards gigabit cell based ATM networks require new adequate transport systems. Integrated designs of transport services, protocol architecture, and implementation features are increasingly needed for the stringent requirements of forthcoming applications on high-speed cell based ATM networks. Explicit support of parallelism can help in performance issues as well as in service flexibility. The transport subsystem PATROCLOS (parallel transport subsystem for cell based high speed networks) is designed with special emphasis on a high degree of inherent parallelism. This allows efficient implementations on multiprocessor architectures such as transputer networks combined with specialised hardware for very time critical functions.

Keyword Codes: C.2.2; C.2.0; C.1.2
Keywords: Network Protocols; Computer-Communication Networks; Multiple Data Stream Architectures (Multiprocessors)

1. INTRODUCTION

During the last years, the characteristics of communication networks changed dramatically. Data rates in the gigabit range are a reality. Furthermore, single networks provide an increasing variety of services. However, there is still a lack of adequate transport systems delivering these features to the applications. A *transport system* combines the functions of the four lower layers according to the OSI Reference Model. Consequently, we use the term *transport subsystem* for systems that cover only subsets of a complete transport system. Transport subsystem entities use *protocols* for communication.

Generally, the use of parallelism is desirable to increase the performance of transport subsystems [1]. Moreover, this can support the flexibility of future subsystems [2]. Their parallelism should be based on a relatively fine granularity to support efficient mappings of transport system protocols on multiprocessor architectures. Furthermore, low overhead should be introduced by the parallelism (e.g., communication overhead among parallel building blocks).

Multiple parallel protocol implementations have already shown possible performance advantages [3, 4, 5, 6, etc.]. However, parallelising standard protocols (e.g., OSI or TCP/IP) suffers from their inherent sequential structure. This is one of the main issues addressed by PATROCLOS, which realises a parallel protocol architecture with fine granularity adequate for multi-

processing. PATROCLOS directly supports parallel implementations on hybrid multiprocessor platforms.

In addition, several characteristics of emerging cell based ATM networks influence a proper protocol design for a transport system. A number of protocol mechanisms need to be changed in order to take advantage of the provided network bandwidth [7, 8]. For example, connection management or error control and flow control need to be adapted to keep end-to-end delay low and to achieve high network utilisation. Moreover, since a large amount of data may be travelling between communicating end systems, it is impractical to maintain a consistent view on the state in both systems. The exchange of connection state information piggy-backed in user data can not be recommended for high performance communications. Mechanisms based on exchanges after dedicated requests (e.g., XTP [9]) or periodic exchanges (e.g., SNR [3]) seem to be more suitable.

The ATM standard with its cell based transmission of information also influences protocol design. However, due to the small inter-cell times, most of the functions should be processed on larger data units [10]. Mainly segmentation and reassembly should be performed at cell level. For example, error recovery should not be performed at the cell level. The very low error rate of such networks does not result in a significant advantage of error recovery at cell level compared with error recovery on larger data units [11, 12].

The parallel transport subsystem PATROCLOS presented in this paper attempts to overcome the discussed limitations and to serve cell based high speed networks. It integrates protocol and implementation issues and provides the required functionalities between the media access layer interface and the transport layer interface in a single, modular designed protocol component. It offers a reliable transport service, in contrast to the MAC service provided, e.g., by the adaptation layer AAL type 4 [13] or the DQDB convergence functions [14].

The paper is organised as follows: Section 2 presents our experiences on parallel protocol processing gained with implementations of several transport and network layer protocols on transputer networks. Some guidelines for parallel protocol designs are derived, which provided the framework for the PATROCLOS architecture, which is introduced in section 3. Section 4 focuses on the parallelism inherent in the PATROCLOS design. Finally, section 5 concludes the paper.

2. PARALLEL PROTOCOL PROCESSING

The use of parallelism at the point of bottleneck is a suitable approach to improve the performance of computing systems in general [15]. Considering processing speeds and memory bandwidth as major bottlenecks for high performance communication subsystems, the use of multiprocessor platforms is an adequate approach to increase their performance. However, most of the standardised protocols allow only a low degree of parallelism.

Different levels of granularity have been chosen for several approaches in parallelising standard protocols. Most of them remain at a level that is based on protocol entities or even complete protocol stacks and their configuration in parallel structures [4]. The approach followed in [16] and enhanced to the PATROCLOS design uses a finer granularity with protocol functions as basic building blocks. However, to extract parallelism at this functional level, detailed analysis of those protocols is required [17].

2.1. Parallel XTP

A sophisticated protocol example supporting parallelism is XTP [9]. Parallelism has been considered in the design process and has resulted in the creation of multiple concurrent finite state machines (FSMs) [18]. With regard to a parallel implementation, XTP offers the following ***advantages*** over standard protocols [19]:

- Integration of transport and network layer
  This integration - often called transfer layer - generally reduces overhead introduced by layered communication architectures. Redundant functionalities being processed in several layers are avoided. It helps supporting parallelism, because it allows to overcome the pipelined structure imposed by hierarchical layers. Thus, a higher degree of concurrent processing on a single packet can be achieved.
- Parallel FSMs
  The design of XTP with concurrent FSMs provides an excellent basis for parallel implementations. A more or less direct mapping on a multiprocessor platform is possible without extensive preceding analysis and decomposition phases.
- Sender and receiver decomposition
  In addition to the parallel FSMs, XTP clearly separates between send and receive functionalities and, thus, permits another degree of parallelism. Both protocol parts can be implemented almost independently of each other. This is very important for the handling of duplex traffic. Duplex connection processing is further simplified by the decomposition of a connection into two independent contexts. Parallel processing of the two data flow directions of a duplex connection is possible because the data streams are handled highly independently of each other.
- Separation of control and information packets
  The distinction of information packets, which contain mainly user data, and control packets, which contain control information (e.g., for flow control or acknowledgement) simplifies parallel processing of user data and control data. The XTP specification provides a single FSM for control packet generation, which can be performed in parallel to user data processing functions.
- Fixed packet format
  The fixed packet format allows processing of packet fields in parallel because the locations of the single packet fields are independent of each other. In earlier versions of XTP, packets consisted of a header and a trailer. In this case, the location of the trailer of a information packet depends on the length of user data. In the latest XTP version 3.6 [9], the trailer consists only of a checksum reducing dependences.

All these features are extremely helpful in designing and implementing a high performance parallel XTP on multiprocessor architectures such as transputer networks. Our experiences with various parallel protocol implementations, including those of OSI protocols, have confirmed this [20]. The XTP implementation based on parallel FSMs [5] is able to process send and receive part in parallel without significant interactions. Parallel processing of receive functions leads to higher performance gains than parallel processing of send functions, because of the mainly uni-directional data flow at receive side. Processing of information and control packets can be performed partly in parallel. Communication overhead caused by event signalling among the FSMs and by context information updates is the major limiting factor.

However, the experiences have also shown that the XTP specification still includes some ***drawbacks*** hindering more efficient parallel software implementations. It should be emphasised that XTP has been designed with a VLSI implementation in mind. Nevertheless, because of

cost reasons, software implementations will also be important. Therefore, we discuss in the following paragraphs some aspects of the specification, which are mainly concerned with parallel software implementations. However, they can also be helpful in improving VLSI implementations.

- Phase dependent activities
  Most of the XTP FSMs are only partly active during different phases of a connection. Thus, they basically improve parallelism between multiple connections during connection establishment/termination and data transfer phases. For example, only 6 out of 11 XTP FSMs are involved in processing during the data transfer phase.
- Load balancing among FSMs
  During the data transfer phase, mainly two FSMs (one FSM on send side and one on receive side) are involved in protocol processing. The other FSMs are only slightly loaded. Therefore, the highest parallelism at the data transfer phase can be achieved by concurrent processing of send and receive functions. The most important FSM of the send part formats outgoing information packets, performs the sender's flow control activities, is involved in rate control, and requests control packets from the peer entity. On the receive side, the main FSM processes all incoming packets. It is involved in processing information as well as control packets, and triggers actions that depend on the checks on received packets.
- Tightly coupled information and control processing
  The use of different types for information packets and control packets is not really reflected in FSM processing. E.g., the main FSM on receive side checks all incoming packets independent of the type, checks user data, and performs control tasks such as flow and rate control, as well as acknowledgement functions. It triggers control packet generation of other FSMs based on results of the checks on incoming packets.
- High interdependences between FSMs
  Bi-directional message flow between some FSMs restricts parallel processing considerably. Some FSMs can only work sequentially, but not in parallel. They send messages to other FSMs and, then, have to wait for the response. Moreover, the co-operation among FSMs is performed by signalling events and by sharing common data bases. The context informations for the connections, also called the context variable list, form the most important data base. It includes the complete state information of a connection and most of the FSMs need to access to at least parts of the informations. The concurrent access of different FSMs sequentialises their processing.

2.2. Design guidelines for parallel protocol architectures

Based on the analysis of XTP FSMs and the performance results of the corresponding multi-processor implementation [5], we derived the following guidelines for protocol architectures appropriate for parallel processing:

- *Reducing data dependences*
  Data dependences among protocol functions hinders parallel processing considerably. Some protocol functions are directly data dependent, i.e., their input data is the output data produced by another function. Additionally, semantical dependence can occur, if multiple outputs of different functions have to be combined to calculate the final result (e.g., different concurrent header analysing functions). The results have to be combined for a decision of the correctness and of subsequent processing steps. Fixed header formats enable parallel processing of header analysing functions without complex header parsing. Another proposal

to overcome the problem of data dependences by choosing new protocol functions with less or without data dependences is discussed in [10].

- *Decoupled control and user data processing*
  Protocol functions can be classified into data transfer functions, which process service data units, and transfer control functions, which control and manage data transfer, e.g., by connection management, and error control [21]. In standard protocols, control and data processing functions are tightly coupled. In that case, processing of user data usually forms the critical path. A more loose structuring permits a higher degree of parallelism by separating control functions from user data processing. Independent FSMs for each type of packet can avoid restrictions for parallelisation as detected in XTP and can allow strict separation of user data and control processing.
  The header fields in user data packets for selection or triggering of control modes or control functions such as acknowledgement do not seem to be appropriate to support the proposed strict separation of user data and control packet processing. Therefore, we propose a scheme, in which user data packets contain absolutely necessary fields only, to offload as many control functions as possible from user data processing. Furthermore, acknowledgement and retransmission of user data should be separated from sending and receiving user data.
  The control functions are often triggered by data transfer, e.g., flow control parameters are frequently exchanged for each data packet. The exchange of control packets can be decoupled from data transfer by explicit requests for control packets, or by periodic state exchange. This is suitable for high speed networks because of the different connection state views of sender and receiver.
- *Modular design*
  Control functions should be processed concurrently to avoid the emergence of a new bottleneck. This is even more important since the calculations for complex flow control and congestion control strategies are becoming increasingly important in high-speed networks. The ability of parallel processing requires an orthogonal design of control functions with very few interactions, which can be achieved by a clear subdivision into different protocol functions (e.g., separation of acknowledgement from rate control). The subdivision allows for a modular design, which also supports a flexible protocol configuration as introduced in [2]. Another approach for a modular protocol design is presented in [22].
- *Reducing event signalling*
  In order to avoid communication overhead between parallel protocol processes, event signalling should be reduced to a minimum. It is often not necessary to signal events for each received packet. Periodic signalling with low period times may also be sufficient.
- *Minimising common resources*
  The functions performed concurrently should avoid the use of any shared resources (e.g., common control packets for multiple control functions) to avoid consistency problems and access conflicts. Therefore, the control functions of a protocol should operate on independent state variables and should also use separate control packets. The decomposition of a protocol into modular units should be done with consideration of the possibility to distribute connection state information. The necessity of globally accessible connection state information reduces parallelism significantly. If a complete distribution of context data can not be avoided, event signalling can be combined with the exchange of connection state information, because frequently, changes in connection state information become important with a new event.

3. PATROCLOS ARCHITECTURE

A protocol suitable for high performance communications should permit an easy mapping on multiprocessor architectures. Moreover, a high degree on flexibility at the service interface is desirable. This issue is not discussed in this paper; related work is presented in [2]. However, both design issues require a new communication model, preferably not based on a layered architecture since this adds considerable overhead, e.g., due to inter-layer communications, the required layer presence -which does not allow the absence of any intermediate layer-, and the layer transparency. The latter leads to redundant placement of protocol functions in several layers, e.g., segmentation and reassembly functions in layer 3 and 4 of the OSI reference model. Furthermore, complex sublayering is used to integrate new networks in established layered communication architectures. For example, an integration of ATM or DQDB networks requires additional adaptation and convergence layers below the MAC service level.

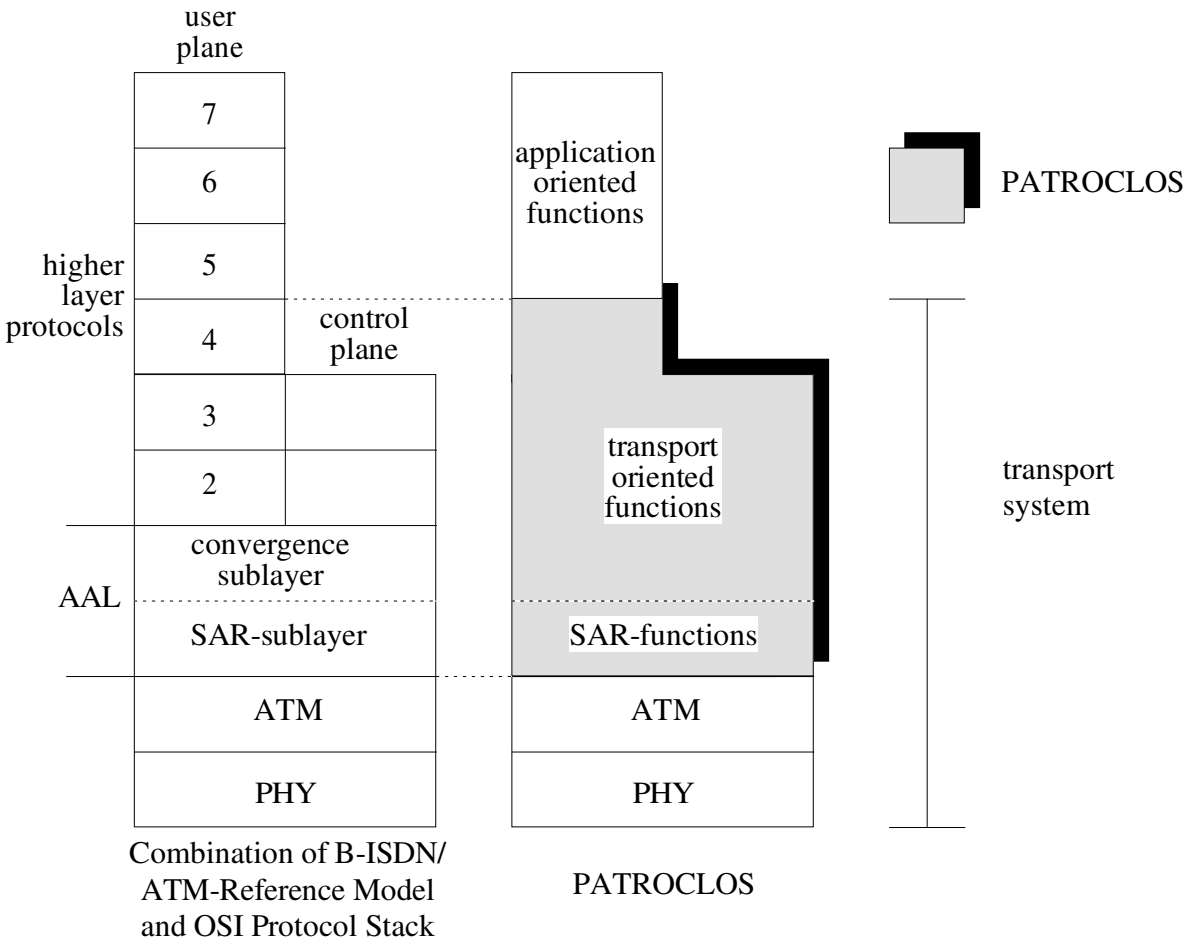

Figure 1: Relationship of PATROCLOS to the B-ISDN/ATM Reference Model

Based on the experiences previously gained with parallel protocol implementations, a parallel transport subsystem for cell based networks, PATROCLOS, has been developed. Another important design issue is the ability of PATROCLOS to work efficiently over cell based high speed networks. PATROCLOS provides a reliable transport service by covering all functions

that usually are performed between media access and the transport service interface. Its relation to the B-ISDN/ATM Reference Model [23] and the OSI Reference Model is shown in Figure 1.

PATROCLOS operates on PATROCLOS-frames (P-frames) and cells only. P-frames form the logical units within the transport subsystem for user data transfer, control functions, and error recovery functions. P-frames can be compared with protocol data units. At the interface to the transport user, transport service data units (TSDUs), consisting of P-frames with additional information, are exchanged. Segmentation and reassembly (SAR) as well as other size changing functions such as multiplexing are only performed at the interfaces of the transport subsystem.

PATROCLOS is mainly designed around a set of independent protocol functions, which are classified as depicted in Figure 2. This allows a high degree of concurrent processing. There is a high degree of independence among functions involved in send and receive processing. Generally, transfer control and data transfer functions are separated in order to decouple control functions from data processing. The data transfer functions are further subdivided into data access and data control functions.

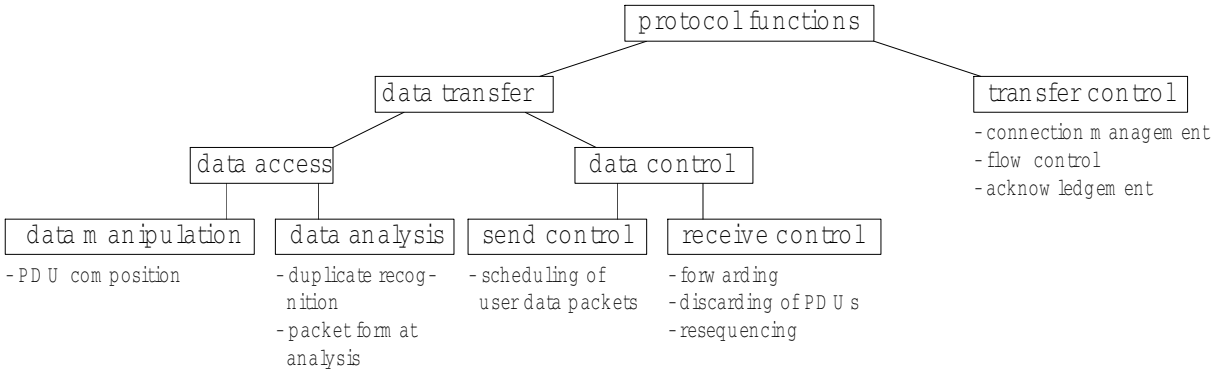

Figure 2 : Protocol functions

Data access functions process user data units and can be divided into the subclasses *data manipulation functions* and *data analysis functions*. Most of the data manipulation functions are located at the send part in contrast to the data analysis functions that are mainly located at the receive side of the transport subsystem.

Data control functions do not manipulate or analyse user data units. They decide the actions to be performed on data units. Data control functions can also be divided into two subclasses called *send control functions* and *receive control functions*.

Transfer control functions (connection establishment and termination, connection data management, flow control, congestion control, rate control, acknowledgement, synchronisation message exchange, and route management) do not operate on user data as the functions described above. They control the connection and its user data flow and exchange control information for control and management purposes.

The logical architecture of PATROCLOS is presented in Figure 3. On the send side, TSDUs are received via the transport service interface and may be segmented. Internally, P-frames are generated by the data manipulation functions or by the transfer control functions respectively. The send control functions determine the next P-frame to be delivered to the net-

work, considering connection states and quality-of-service parameters. At this point, segmentation of P-frames may be required. P-frames can be distinguished in P-data-frames for user data and P-control-frames for control information.

On the receive side, arriving cells are reassembled to P-frames, which are delivered to the data analysis functions or to the transfer control functions. The data analysis functions check for errors on user data such as duplicates, lifetime errors, etc. The results are delivered to the transfer control functions or to the receive control functions.

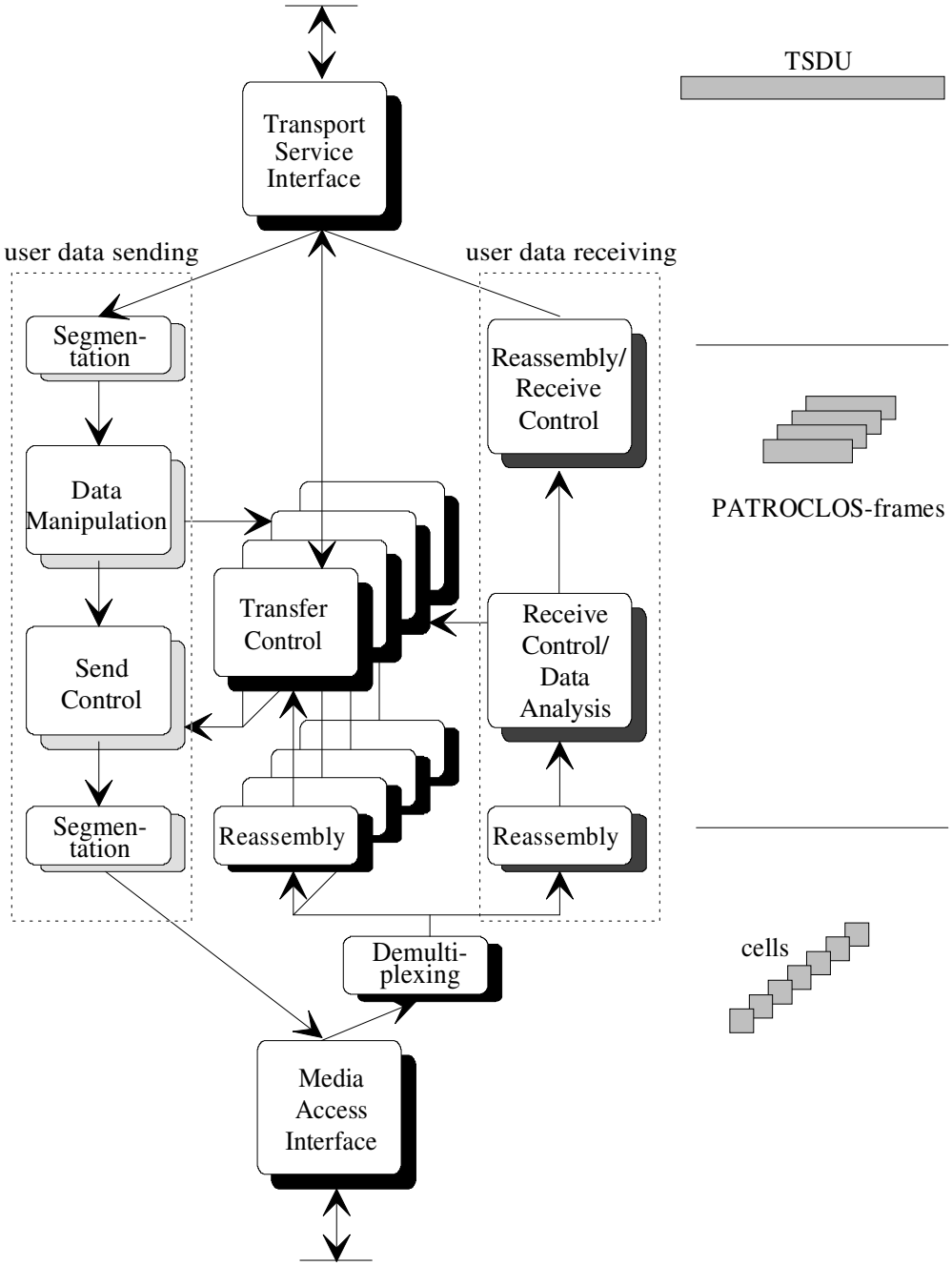

Figure 3 : Architecture of PATROCLOS

The transfer control functions update connection data and perform functions like acknowledgement or flow control dependent on the results of the analysis. The receive control functions decide the actions to perform with the P-data-frame, e.g., forwarding, discarding, or delivering P-data-frames to the upper layer considering delay jitter requirements. If needed, P-data-frames are reassembled before delivery to the transport system user.

4. PATROCLOS INHERENT PARALLELISM

The protocol functions of PATROCLOS are performed by multiple parallel FSMs. FSMs belonging to the same PATROCLOS entity exchange messages for synchronisation and communication purposes. This internal event signalling is performed periodically to reduce processing overhead, if possible. In the remaining of this section, the FSMs as well as the communication and co-operation among them are presented in more detail.

4.1. Parallel FSMs

PATROCLOS consists of two types of FSMs:

- *Interface FSMs* for interactions with neighboured layers, which are located at the interfaces of the transport subsystem.
- *Protocol FSMs* communicating by separate protocols with peer FSMs at the peer entity.

4.1.1. Interface FSMs

The interface FSMs support interactions of PATROCLOS with the transport service user or with the underlying network unit. They are only involved in local communications inside a system, and they do not have to communicate with FSMs at the peer entity. The transport interface FSM exchanges TSDUs with the transport service user. The FSM triggers actions of the protocol FSMs depending on the received TSDUs, delivers TSDUs to the transport service user, and is involved in performing secure connection termination. The network interface FSM controls transmission of P-frames to the underlying network unit.

4.1.2. Protocol FSMs

The transfer control functions are performed by the protocol FSMs. PATROCLOS consists of protocol FSMs for all transfer control functions and for outgoing and incoming user data (cf. Figure 4). They are designed to decouple the connection state information exchange from user data exchange effectively. Acknowledgements and connection state information exchange are triggered by expired timers, explicit requests, or error situations. Protocol FSMs communicate directly with the corresponding peer FSMs. Each communication relation has its own protocol and P-frames. Multiplexing of P-frames by different FSMs is avoided to support parallelism. The FSM protocols have their individual error recovery mechanisms and timers. The decomposition into independent FSMs not only permits parallel execution but also has the benefit of a highly modular system, which can be configured easily depending on the requirements of the applications and the characteristics of the networks, strategies, mechanisms and parameters. Configuration issues of transport systems are covered in [2].

- *Connection management*
  A single connection management FSM manages the complete duplex connection. It supports establishment, monitoring, maintenance, and correct termination of connections. Handshake-based or implicit establishment mechanisms are provided according to the requested service. If required, the FSM performs signalling tasks. The peer FSMs exchange P-control-frames for connection establishment. They initialise the connection and forward quality-of-service requirements to other FSMs. The FSMs also guarantee orderly connec-

tion termination and the unique use of connection identifiers. Further division of the connection management FSM is not performed because establishment, maintenance, and termination of connections occur sequentially.

- *User data transfer*
  User data transfer FSMs send and receive user data. The main tasks of the two FSMs are formatting and analysing P-data-frames.

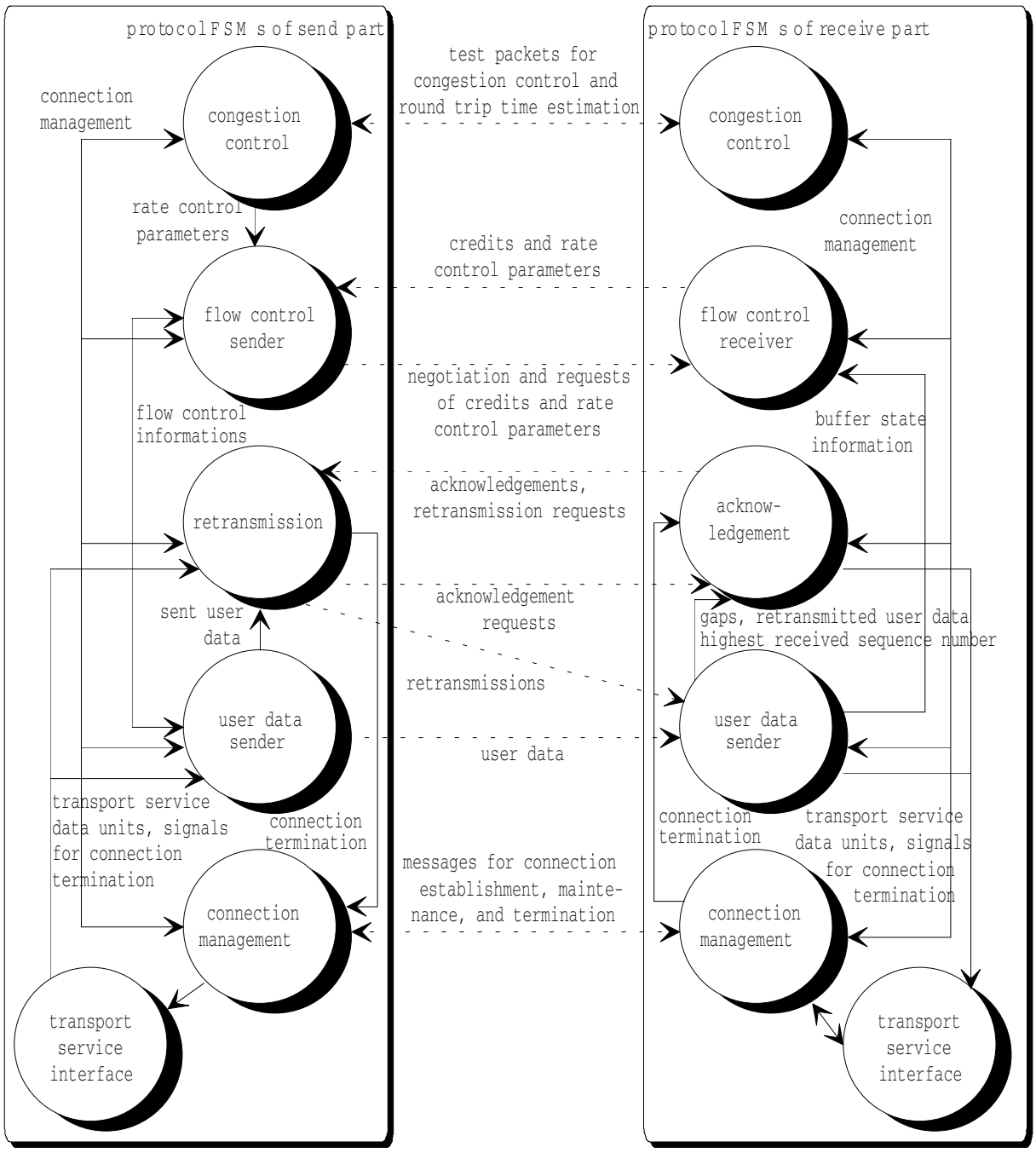

Figure 4: Co-operation of FSMs

- *Acknowledgement and retransmissions*
  The acknowledgement/retransmission functions can be divided into two FSMs. The FSM on the send side (retransmission FSM) receives acknowledgements from the peer's FSM on the receive side (acknowledgement FSM). The retransmission FSM may also request acknowledgements, or send the state of acknowledged data to the peer's acknowledgement FSM. Retransmissions are performed depending on the acknowledgement information and are sent to the peer's user data receive FSM. The retransmission strategy and the management of unacknowledged data is implemented in this FSM. The retransmission FSM controls also buffers of unacknowledged user data and has to release buffer space of acknowledged data. The acknowledgement FSM sends negative or positive acknowledgements to the peer's retransmission FSM and requests retransmissions based on the results received from the user data receive functions.
- *Routing and relaying*
  The FSM performs functions required in intermediate systems such as relaying, routing, route management, and monitoring.
- *Flow and rate control*
  For flow and rate control, there are two FSMs for each connection, i.e., one FSM for each transfer direction. The FSMs control data flow by the interpretation of the peer entity's feedback signals, e.g., window and buffer sizes, and rate values. For rate control, both peer FSMs negotiate appropriate rate parameters. The flow control receive FSM can transmit rate parameters or credit values according to the current traffic and available buffer space. The peer's flow control send FSM triggers send control functions of the same entity by signalling parameters to the send control FSM. It may also request new credit values from the peer's flow control receive FSM.
- *Congestion control*
  The congestion control FSMs control data flow by the interpretation of feedback signals from the network or the peer entity, e.g., congestion indications, round trip times, and inter-arrival times of test packets. The FSMs may exchange test P-frames to identify congestions or to estimate round trip times. Based on test packet delays and inter-arrival times, congestion situations can be detected and the rate can be decreased [24]. The FSM may also react to the feedback signals of the network nodes.
- *Delay jitter control*
  The jitter control FSMs support the timing relationships between sent and received data. Several applications require well defined data packet inter-arrival times. These FSMs send and receive time stamps for transferred user data. The receive FSM controls correct delivery of user data to the transport system user considering the received time stamps.

4.2. Communication among protocol FSMs

The P-frame for all FSMs consists of a basic header (cf. Figure 5a), FSM specific fields (data), and an optional checksum. The basic header contains addresses, identifiers, type fields, and lifetime values.

The FSMs are addressed by entity address extensions (*fsm*-field). The extensions are simple bit maps, and each bit represents a certain FSM. This scheme allows multicast addressing of protocol FSMs. This can be useful if a FSM of one entity has to send the same data to several peer entity FSMs. In that case, the data are transmitted once and are distributed at the peer entity to all destination FSMs. This mechanism is used for implicit connection establishment. The first P-data-frame of a connection is distributed to the connection management FSM and

to the user data receive FSM. The *type*-field allows to distinguish P-frames used by a single FSM.

The FSM specific fields for P-data-frames (cf. Figure 5b) consist of a fixed FSM specific header (length informations and sequence numbers) and user data of variable length (*data*-field). By using the IP addressing scheme, the header (basic and specific header) of a P-data-frame fits exactly in one AAL type 4 cell. P-control-frames have usually fixed sizes of only one or two cells to be appropriate for cell based networks. In networks with larger transmission units, e.g., FDDI, multiple P-control-frames can be concatenated.

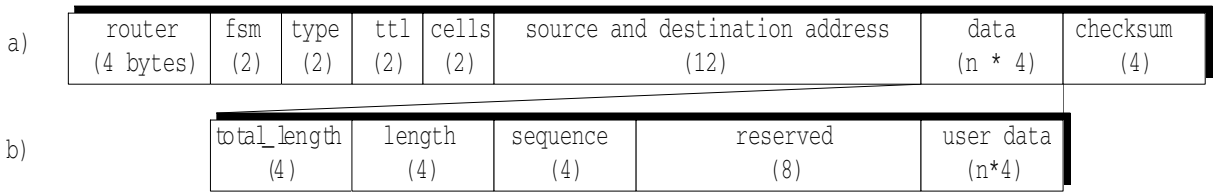

Figure 5 : a) Basic P-frame format, b) P-data-frame format

Most of the P-control-frames fit within a single ATM cell and do not introduce significant overhead on the network. Furthermore, the short P-control-frames are usually sent less frequently than user data. The PATROCLOS design tries to find a trade-off between additional bandwidth required for the transmission of control information and increased processing performance. Especially the latter is critical for very high performance communication subsystems.

4.3. Co-operation among FSMs of one entity

The co-operation among FSMs is illustrated in Figure 4. The system is designed to minimise interactions among FSMs of the same entity. Furthermore, communication among FSMs is uni-directional to allow a high degree of parallelism. Recall that in XTP bi-directional communications reduce the achievable parallelism considerably.

Connection establishment and termination

- The connection management FSM is responsible for connection initialisation and correct release. Therefore, this FSM has to activate and initialise the other FSMs of the same entity for a new connection and has also to deactivate these FSMs after connection termination.
- The connection management FSM has to ensure by co-operation with the retransmission FSM and the transport interface FSM that all data, which are delivered before initiating connection termination, are sent and acknowledged.

Flow Control

- The flow control state FSM on send side controls the user data send FSM by the negotiated flow control parameters. The user data FSM can request new credit values if required.
- On receive side, the user data receive FSM has to signal buffer state information to the flow control FSM, which calculates new parameters for the peer's send flow control FSM.

Retransmissions and Acknowledgements

- The user data send FSM gives control of user data previously sent to the retransmission FSM, which is responsible for further retransmissions to the peer's user data receive FSM. Retransmissions depend on incoming acknowledgements from the peer's acknowledgement FSM.
- On receive side, the user data receive FSM signals information about received data to the acknowledgement FSM periodically with short time intervals.

5. PROTOTYPE IMPLEMENTATION

PATROCLOS is appropriate but not limited to be implemented on a hybrid multiprocessor architecture [25]. Parallel architectures for communication subsystems are successfully built with transputer networks [5, 6, 16]. The FSMs have been implemented as processes, and each process may be mapped to a single processor of a suitable multiprocessor configuration. The processes are interconnected by logical channels, which are mapped to software or hardware channels. Processes on the same processor are interconnected by software channels, processes on different processors are interconnected by hardware channels. Processes of the same entity communicate with each other via message passing across channels or via common data in shared memories. Global memories for user data are realised by multi-port memories.

Simple protocol functions on cell level (e.g., SAR processing according to AAL type 4) have been implemented by dedicated hardware processing units to support the high network bandwidth. Currently, memory management tasks are distributed among the processors of the send and receive pipeline. Timer processing may be performed by separate processors. However, for timer processing, memory management and further time-critical functions like checksum calculation specialised hardware units will substitute software implementations. The combination of universal processors and specialised processors results in a hybrid multiprocessor architecture.

The implementation architecture is depicted in Figure 6. It will be connected via a DMA unit to a host system. On the send side, TSDUs are segmented by the segmentation processor and delivered to the data manipulation processor. The use of shared memories avoids copying of data between processors. The send control processor receives P-frames from the data manipulation processor and from the transfer control processors, determines the next P-frame to be sent, and delivers corresponding P-frame informations to the segmentation hardware, which performs cell segmentation. Cell headers and trailers are exclusively generated by the segmentation hardware.

On the receive side, the reassembly hardware units reassemble incoming cells to P-frames. A demultiplexing hardware performs demultiplexing below the reassembly level. The incoming cells are distributed to the reassembly hardware units of the data manipulation processor and the transfer control processor modules. The reassembly hardware has to signal completely received P-data-frames and errors (e.g., bit errors, cell loss, and missequenced cells). The received P-data-frames are delivered via the receive control processor to the reassembly processor, which generates TSDUs. The TSDUs are delivered via the transport service interface to the transport service user.

6. CONCLUSIONS

In this paper, PATROCLOS, a parallel transport subsystem design appropriate for high performance communication subsystems over cell based ATM networks, has been presented. The general goal was an integrated design of protocol and implementation architecture issues. The aspect of integration is a driving factor towards high performance communication platforms. Multiprocessor systems may be the appropriate candidates for future gigabit rate networks and, thus, a protocol design appropriate for efficient implementation architectures is required.

The presented design is highly influenced on experiences gained during the parallel implementation of XTP on transputer networks. Some general guidelines for parallel protocol design

have been derived from that work. The key issue is the decomposition of a protocol entity into concurrently executable FSMs. Therefore, the protocol consists of multiple modular FSMs operating almost independently from each other. The design comprises a separate FSM for each control function and for user data processing. With respect to concurrent processing, synchronisation among FSMs of a single protocol entity is reduced by associating each FSM with an individual protocol for message exchange between FSMs residing in different entities.

A prototype version of PATROCLOS is currently being implemented on a hybrid multiprocessor architecture using transputers as general purpose processors and dedicated hardware for time-critical protocol functions. Preliminary performance evaluations show significant improvements compared to similar implementations of other protocols such as XTP.

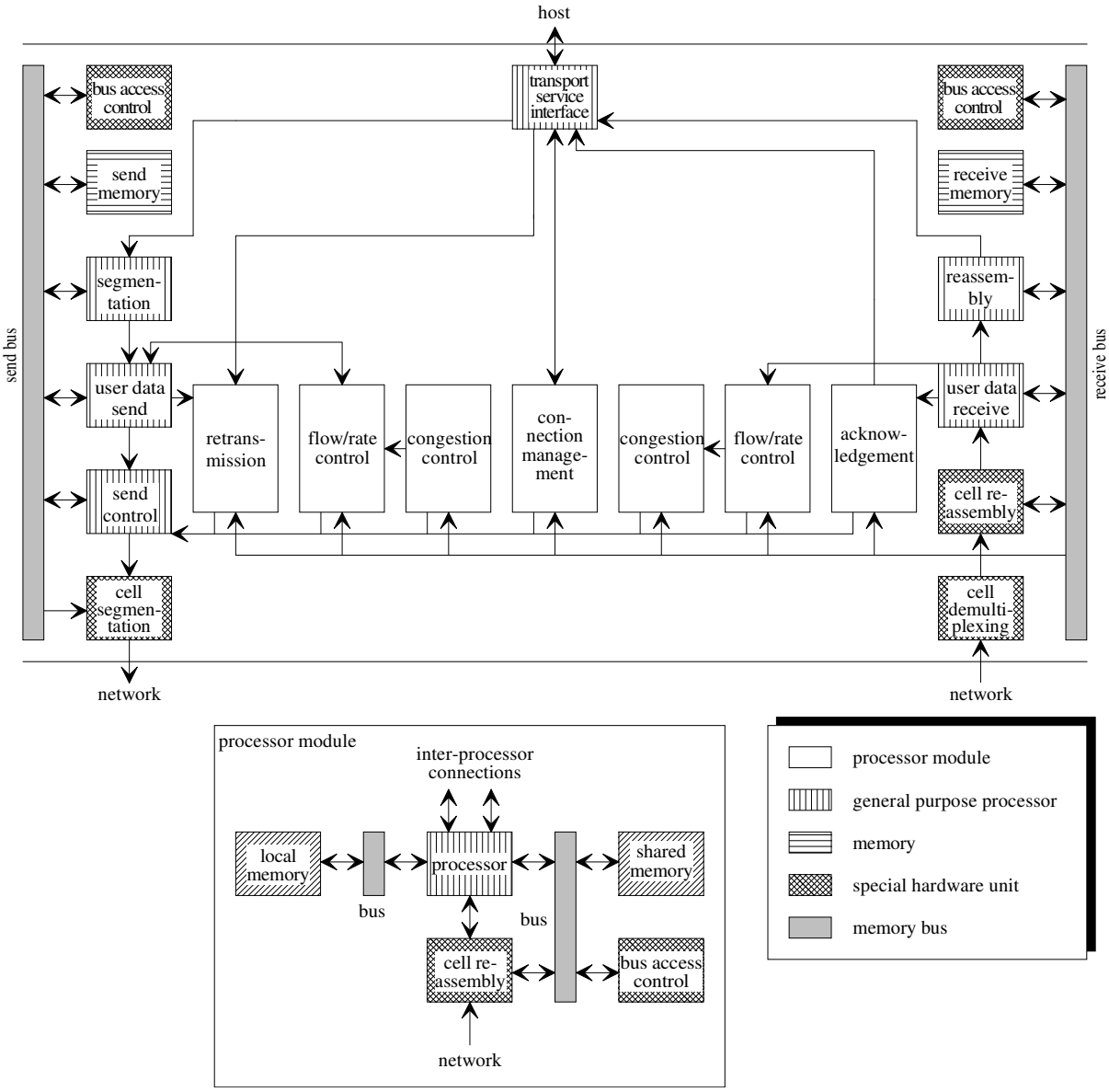

Figure 6: Implementation architecture

REFERENCES

[1] Zitterbart, M.
Parallelism in Communication Subsystems
to appear in: Tantawy, A. (ed.): High Performance Communications, Kluwer Academic Publisher, 1993

[2] Zitterbart, M.; Stiller, B.; Tantawy, A.
A Model for Flexible High-Performance Communication Subsystems
to appear in: Journal of Selected Areas in Communication, 1993

[3] Sabnani, K.; Netravali, A.; Roome, W.
Design and Implementation of a High Speed Transport Protocol
IEEE Transactions on Communications, Vol. 38, No. 11, November 1990, pp. 2010-2024

[4] Jain, N.; Schwartz, M.; Bashkow, T.R.
Transport Protocol Processing at Gbps Rates
Proceedings of the ACM SIGCOMM ´90, September 24-27, 1990, Philadelphia, USA, pp. 188-199

[5] Braun, T.; Zitterbart, M.
A Parallel Implementation of XTP on Transputers
16th Annual IEEE Conference on Local Computer Networks, October 14-17, 1991, Minneapolis, USA, pp. 172-179

[6] Kaiserswerth, M.
A Parallel Implementation of the ISO 8802-2.2 Protocol
IEEE TriComm '91, Chapel Hill, NC, USA.

[7] La Porta, T.F.; Schwartz, M.
Architectures, Features, and Implementation of High-Speed Protocols
IEEE Network, Vol. 5, No. 3, May 1991, pp. 14-22

[8] Doeringer, W.; Dykeman, D.; Kaiserswerth, M.; Meister, B.W.; Rudin, H.; Williamson, R.
A Survey of Light-Weight Transport Protocols for High-Speed Networks
IEEE Transactions on Communications, Vol. 38, No.11, November 1990, pp. 2025-2039

[9] Strayer, W.T.; Dempsey, B.J.; Weaver, A.C.
XTP: The Xpress Transfer Protocol
Addison-Wesley Publishing Company, 1992

[10] Feldmeier, D.; McAuley, A.
Reducing Ordering Constraints to Improve Performance
Proceedings of the 3rd IFIP WG6.1/6.4 Workshop on Protocols for High-Speed Networks, May 13-15, 1992, Stockholm-Kista, Sweden, pp. 3-16

[11] Lai, W.S.
Recovering from Congestion Losses in High-Speed Networks
in: Johnson, M. J. (ed.): Protocols for High-Speed Networks, II, North-Holland, 1991, pp. 17-23

[12] Greene, D.H.; Lyles, J.B.
Reliability of Adaptation Layers
Proceedings of the 3rd IFIP WG6.1/6.4 Workshop on Protocols for High-Speed Networks, May 13-15, 1992, Stockholm-Kista, Sweden, pp. 195-205

[13] CCITT Recommendation I.363
B-ISDN ATM Adaptation Layer Specification
Matsuyama, December, 1991

[14] Institute of Electrical and Electronical Engineers
IEEE Standards for Local and Metropolitan Area Networks: Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network
IEEE Std. 802.6-1990, May 20, 1991

[15] Stone, H.S.
High Performance Computer Architecture
Addison-Wesley Publishing Company, 1987

[16] Zitterbart, M.
High Speed Transport Components
IEEE Network Magazine, Vol. 5, No. 1, January 1991, pp. 54-63

[17] Koufopavlou, O.G.; Tantawy, A.; Zitterbart; M.
Parallelization of TCP/IP for Very High Speed Networks
17th Annual IEEE Conference on Local Computer Networks, September 13-16, 1992, Minneapolis, USA, pp. 576-585

[18] Protocol Engines Inc.
XTP Protocol Definition
Revision 3.4, July 17, 1989

[19] Braun, T.; Zitterbart, M.
Parallelism in XTP
TRANSFER, Protocol Engines Inc., Vol. 5, No. 3, May/June 1992, pp. 11-15

[20] Zitterbart, M.
Parallel Protocol Implementations on Transputers - Experiences with OSI TP4, OSI CLNP, and XTP
IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems, Tucson, Arizona, U.S.A., February 17-19, 1992

[21] Clark, D.; Tennenhouse, D.
Architectural Considerations for a New Generation of Protocols
ACM SIGCOMM ´90, September 24-27, 1990, Philadelphia, USA, pp. 200-208

[22] Haas, Z.
A Protocol Structure for High-Speed Communication over Broadband-ISDN
IEEE Network Magazine, Vol. 5, No. 1, January 1991, pp. 64-70

[23] CCITT Recommendation I.321
B-ISDN Protocol Reference Model and its Application
Matsuyama, December 1991

[24] Keshav, S.; Agrawala, A.; Singh, S.
Design and Analysis of a Flow Algorithm for a Network of Rate Allocating Servers
in: Johnson, M.J. (ed.): Protocols for High-Speed Networks, II, North-Holland, 1991, pp. 55-72

[25] Schmidt, C.
Design of a Transport System Based on Parallel Finite State Machines
Diploma thesis (in German), University of Karlsruhe, July 1992